

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **A computer vision approach to drone-based traffic analysis of road intersections**

**Gustavo Ramos Lira**



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Daniel Moura

Co-supervisor: Prof. Rosaldo Rossetti

July 26, 2015



# **A computer vision approach to drone-based traffic analysis of road intersections**

**Gustavo Ramos Lira**

Mestrado Integrado em Engenharia Informática e Computação

July 26, 2015





# Abstract

In recent years, there has been interest in detailed monitoring of road traffic, particularly in intersections, in order to obtain a statistical model of the flow of vehicles through them. While conventional methods – sensors at each of the intersection’s entrances/exits – allow for counting, they are limited in the sense that it is impossible to track a vehicle from origin to destination. This data is invaluable to understand how the dynamic of a city’s mobility works, and how it can be improved. Therefore, new techniques must be developed which provide that kind of information. One of the possible approaches to this problem is to analyse video footage of said intersections by means of computer vision algorithms, in order to identify and track individual vehicles. One of the possible ways to obtain this footage is by flying a drone – a small unmanned air vehicle (UAV) – carrying a camera over an intersection.

Some work has been done with this solution in mind, but the usage of a top-down birds-eye perspective, obtained by flying the drone directly above the intersection, rather than at an angle, is limited or inexistent. This approach is interesting because it circumvents the problem of occlusions present in other footage capture set ups. The focus of this dissertation is, then, to develop and apply computer vision algorithms to footage obtained in this way in order to identify and track vehicles across intersections, so that a statistical model may be extracted. This model is based on said association of an origin and a destination. The algorithm employs background subtraction and vehicle tracking with Kalman filters. The most significant challenge was how to compensate for camera movement. Based on the implementation which was developed, this the approach proved useful for tracking cars, buses and trucks under some conditions.



# Resumo

Recentemente, tem existido interesse na monitorização do tráfego automóvel, particularmente em cruzamentos, com o objectivo de obter um modelo estatístico do fluxo de veículos através dos ditos. Enquanto que os métodos convencionais - sensores em cada uma das entradas/saídas do cruzamento - permitem contar o número de veículos, estes são limitados, dado que não permitem acompanhar um veículo da entrada à saída. Estes dados são preciosos para perceber como funciona a dinâmica de mobilidade de uma cidade, e como pode ser melhorada, pelo que novas técnicas que forneçam esse tipo de informação têm que ser desenvolvidas. Uma das abordagens possíveis a este problema é a análise, por meio de algoritmos de visão por computador, de vídeo capturado dum cruzamento, para identificar e seguir veículos. Uma das formas possíveis de obtenção de vídeo é a utilização de um "drone" - um pequeno veículo aéreo não tripulado - com uma câmara, para voar por cima de um cruzamento e o filmar.

Algum trabalho foi feito com esta ideia em mente, mas a utilização de uma perspectiva vertical *birds-eye*, em vez de uma perspectiva inclinada, é limitada ou inexistente. Esta abordagem é interessante porque contraria o problema das oclusões patente noutras formas de captura de imagem. O objectivo desta dissertação é, então, desenvolver e aplicar algoritmos de visão por computador a imagens obtidas desta maneira, para identificar e seguir veículos num cruzamento, para que um modelo estatístico do dito possa ser obtido. Este modelo baseia-se na supracitada associação de entradas a saídas. A solução desenvolvida passa pela utilização de *background subtraction* e de acompanhamento de veículos usando filtros de Kalman. Com base na implementação que foi desenvolvida, esta abordagem parece ser útil para carros, autocarros e outros veículos semelhantes, sob determinadas condições.



# Acknowledgements

I would like to thank my supervisors, Professor Daniel Moura and Professor Rosaldo Rossetti, for their work, dedication and invaluable advice, without which this dissertation would not exist.

I would also like to thank all of the faculty members who have motivated me and taught me over the past five years.

A word of appreciation to my friend Paulo Alcino for letting me bounce ideas off of him when I was stuck and for providing some spectacularly useful ideas himself.

I thank my parents, my friends and Sara for their support.

Gustavo Ramos Lira



*“Engineering stimulates the mind. Kids get bored easily. They have got to get out and get their hands dirty: make things, dismantle things, fix things. When the schools can offer that, you’ll have an engineer for life.”*

Bruce Dickinson





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Motivation . . . . .	2
1.3	Goals . . . . .	2
1.4	Document structure . . . . .	3
<b>2</b>	<b>State of the art</b>	<b>5</b>
2.1	Computer Vision . . . . .	5
2.1.1	Feature detection . . . . .	5
2.1.2	Feature tracking . . . . .	9
2.1.3	Background subtraction . . . . .	10
2.1.4	Morphological operations . . . . .	12
2.1.5	Overview . . . . .	13
2.2	Traffic monitoring . . . . .	13
2.2.1	Road traffic monitoring solutions . . . . .	13
2.2.2	Models . . . . .	15
<b>3</b>	<b>Solution</b>	<b>17</b>
3.1	Overview . . . . .	17
3.2	Development process . . . . .	17
3.3	Technical details . . . . .	20
3.3.1	Input data . . . . .	20
3.3.2	Output data . . . . .	20
3.3.3	Configuration file format . . . . .	20
3.3.4	Algorithm parameters . . . . .	20
3.4	Limitations . . . . .	22
3.5	Implementation . . . . .	22
<b>4</b>	<b>Experimental results</b>	<b>27</b>
4.1	Metrics . . . . .	27
4.2	Setup and test cases . . . . .	28
4.2.1	Source 1 . . . . .	28
4.2.2	Source 2 . . . . .	29
4.3	Characterization of the clips . . . . .	29
4.4	Results . . . . .	31
4.4.1	Clip 1 . . . . .	31
4.4.2	Clip 2 . . . . .	33
4.4.3	Clip 3 . . . . .	33

## CONTENTS

4.4.4	Clip 4 . . . . .	35
4.4.5	Clip 5 . . . . .	36
4.5	Analysis of results . . . . .	37
<b>5</b>	<b>Conclusion</b>	<b>41</b>
	<b>References</b>	<b>43</b>

# List of Figures

2.1	An example of edge detection [McL10]	7
2.2	An example of corner detection [figa]	8
2.3	An example of blob detection [Guy08]	9
2.4	An example of feature tracking [figb]	10
2.5	Kalman filter being used to track a ball moving from left to right, even when occluded. The point where occlusion ends is marked in red [Kau]	11
2.6	An example of background subtraction. In the lower half, white represents foreground and black represents background [figc]	12
3.1	Flowchart of the general execution of the algorithm	23
3.2	Distortion resulting from attempting to find a transformation matrix (left) and reference frame (right)	24
3.3	Same frame before (top) and after (bottom) blob filtering. Note the disappearance of the small blobs identified in the trees, zebra crossings and hanging wires.	24
3.4	Example of the GUI being used to show each point in several trajectories	25
4.1	Identification of the source 1 intersection limits. 1 - left; 2 - bottom; 3 - top; 4 - right	29
4.2	Identification of the source 2 intersection limits. 1 - top left; 2 - bottom left; 3 - right	30
4.3	A group of motorcycles stopped in the middle of the intersection (highlighted in red)	37
4.4	Difference in level of zoom between clips 2 (top), 3 (middle) and 4/5 (bottom)	39

## LIST OF FIGURES

# List of Tables

4.1	Ground-truth values. Columns represents entrances and rows represent exits . . .	30
4.5	Distribution of traffic given the ground-truth values . . . . .	30
4.9	Ground-truth values. Columns represents entrances and rows represent exits . . .	31
4.12	Distribution of traffic given the ground-truth values . . . . .	31
4.15	Results for clip 1 in 360p . . . . .	32
4.19	Results for clip 1 in 480p . . . . .	32
4.23	Results for clip 1 in 720p . . . . .	32
4.27	Running times for clip 1 . . . . .	32
4.28	Results for clip 2 in 360p . . . . .	33
4.32	Results for clip 2 in 480p . . . . .	33
4.36	Results for clip 2 in 720p . . . . .	33
4.40	Running times for the clip 2 . . . . .	34
4.41	Results for clip 3 in 360p . . . . .	34
4.45	Results for clip 3 in 480p . . . . .	34
4.49	Results for clip 3 in 720p . . . . .	34
4.53	Running times for clip 3 . . . . .	35
4.54	Results for clip 4 in 360p . . . . .	35
4.58	Results for clip 4 in 480p . . . . .	35
4.62	Results for clip 4 in 720p . . . . .	35
4.66	Running times for clip 4 . . . . .	36
4.67	Results for clip 5 in 360p . . . . .	36
4.71	Results for clip 5 in 480p . . . . .	36
4.75	Results for clip 5 in 720p . . . . .	36
4.79	Running times for clip 5 . . . . .	37

## LIST OF TABLES

# Abbreviations

CV	Computer Vision
MOG	Mixture of Gaussians
HDD	Hard disk drive





# Chapter 1

## Introduction

This dissertation concerns itself with the development of a traffic analysis algorithm based on computer vision techniques, used on footage captured vertically above intersections using a drone. This field is of interest given the high value of gathering data about mobility and traffic patterns in an urban environment, which stems from the possibility of using this data to optimize urban traffic control systems such as traffic lights.

There is previous work regarding gathering data about traffic patterns, of which some has been done thanks to the video capture and subsequent analysis, but there no known implementation of a system where a drone is floated vertically over an intersection, and thus there is a gap in the field which can be worked on.

In the current chapter, the context, motivation and goals of the dissertation are presented, as well as an overview of the structure of the rest of the document.

### 1.1 Context

This work is positioned in the fields of computer vision and road traffic monitoring. These two fields intersect when one attempts to monitor road traffic by means of analysis of images captured by some video device positioned relative to the road.

Additionally, this research was done within the context of the Future Cities project, an European project that aims at improving the standard of living of cities' populations by applying information technologies and for which Porto is a living lab.

This work was developed in the Faculty of Engineering of the University of Porto, within the Artificial Intelligence and Computer Science lab.

## 1.2 Motivation

The motivation for this work is threefold: scientific, applicational and technological. Each of these is further detailed below.

The scientific motivation for this work is twofold: perform a gap analysis and experiment with computer vision techniques applied to road traffic monitoring. The first point equates to looking at the state of the art in these fields, seeing where there is work related to both, and then identifying where this work can be improved. For this purpose, a literature review of both computer vision techniques and road monitoring techniques and models has been done. The second point is related to the scientific interest of applying computer vision algorithms to the vertically captured images in order to identify why this situation might present challenges for those algorithms, and how these challenges could be conquered, if at all.

The applicational motivation is, of the three motivations identified, the one most rooted in real-world needs. The analysis of an intersection's dynamic and the resulting data have incredible potential: they allow for optimization of traffic signs' timing, identification of possible bottle necks, prediction of traffic jams and modelling of a cities' road-users behaviour, as well as the seeding of simulators [PRK11] which can then be used to predict and perfect the aforementioned tasks. This integration into simulators can be taken further, if one considers a whole platform dedicated to transportation analysis [ROB07], a concept which is predicated on the idea of Artificial Transportation Systems [RLT11] [RE15].

The technological motivation is based around the idea of associating two previously independent technologies - drones and computer vision - to draw benefits from both and come up with a new system for traffic monitoring as an alternative to more traditional systems, such as road-level sensors.

## 1.3 Goals

The goal of this dissertation project is to design and implement an algorithm capable of identifying individual vehicles, based on computer vision techniques, and of tracking those vehicles through an intersection, to then build a statistical model of said intersection. Additionally, it is also a goal to test this algorithm with real-world footage to prove its reliability.

The project concerns itself with the development of an algorithm which extracts data from images captured from a top-down perspective. This perspective is interesting because it avoids the problem of occlusions, thus potentially facilitating the identification and tracking process. It has not been previously explored, but shows promise given the preliminary results here obtained.

The input images are assumed to be available already, and their means of capture is outside the scope of the research. Additionally, the camera is considered to remain in an almost fixed position above the intersection, with relatively low amounts of shaking. Traffic is assumed to flow unperturbed by jams or accidents and vehicles are considered to enter the intersection and one limit and exit at a different one, so u-turns shall be ignored.

### **1.4 Document structure**

Beyond this introduction, the current document is divided into another 4 chapters. In chapter 2, several computer vision algorithms currently being used are described and analysed, and current road traffic monitoring algorithms are described, with their weaknesses and strong points identified. Chapter 3 describes the solution which was developed, as well as the process which led to it. Chapter 4 shows the experimental results which were obtained. Finally, the conclusion can be found in chapter 5. It provides an overview of what has been analysed, as well as a look into future work.

## Introduction

## Chapter 2

# State of the art

This chapter presents an overview of current techniques in Computer Vision, and current research in traffic monitoring.

### 2.1 Computer Vision

Computer vision concerns itself with the analysis of visual data (both static and video) to extract information and convert it into some numeric form. It is closely related to the fields of AI, machine learning and mathematics. For this last one, specifically the areas of statistics and geometry.

It is commonly used to extract patterns or features from an image, with the intent of identifying objects which can then be counted or tracked, for various purposes. In the context of the current work, computer vision is interesting to analyse video footage of intersections in order to identify and track vehicles.

There are many computer vision techniques, depending on what kind of processing is to be done to the image. This chapter will focus on feature identification, feature tracking, background subtraction and morphological operations, because these four areas are the most prevalent when dealing with traffic monitoring.

There are also several different computer vision libraries which provide some of the algorithms mentioned further on. Some examples are OpenCV [[Ope](#)], VXL [[Pro](#)], IVT [[IVT](#)], AForge.NET [[AFo](#)] and the Matlab Computer Vision System Toolbox [[Mat](#)]. When discussing different algorithms, it will be noted when these are implemented by OpenCV. This is because OpenCV is the library expected to be used during the research phase, given that it is one of the most prevalent and complete ones.

#### 2.1.1 Feature detection

Feature detection is any process by which points of interest (the features) are found on an image. They work by deciding for each pixel if it belongs to a feature or not. What a feature is varies depending on the goal of performing feature detection, because different applications consider different regions "interesting".

Features are usually categorized in 4 different types: edges, corners, blobs and ridges. An edge is a boundary between two regions in an image, and is not necessarily a straight line. A corner is also known as an interest point, and corresponds to some isolated point in the image which differs from its neighbours in some way. Blobs are regions of interest similar to corners but composed of multiple points, rather than just one. Finally, ridges are considered to be sets of connected points where a certain function reaches a local maximum (similar to the geological concept of ridge - a series of elevations which forms an elongated crest). Similarly, a valley can be defined as a set of connected local minima. Further on, the focus will be on the first 3 types, as ridge detection has little to no application in vehicle tracking.

Feature detection is usually one of the first operations in a larger algorithm, so that some regions are isolated and then further analysed with other means. Because of this, it is critical that feature detection algorithms perform well and present good and accurate results. If the features are detected incorrectly, the whole algorithm will, most likely, perform badly.

#### **2.1.1.1 Edge detection**

Edge detection (figure 2.1) aims at finding boundaries which separate areas of a image with significantly different brightness. These are important because they usually correspond an object's boundaries, a change in material or a change in lighting conditions in the source image. Ideally, one wants an edge detector to return a set of connected curves, but in practice this rarely happens because edges have discontinuities and are fragmented. However, this data is still useful and allows for significant reductions in processing time in further steps, because these no longer have to scan the whole image.

Edges are usually defined as sudden changes in brightness. Ziou and Tabbone go into detail regarding the different definitions of edges in [ZT88] but suffice it to say they are identified in one of two ways: either a local maximum in first-order derivative or a zero-crossing in the second-order derivative of the brightness function.

An edge detector can be evaluated according to three criteria defined by John Canny [Can86]

1. Good detection - minimize both false negatives and false positives, which corresponds to maximizing signal-to-noise ratio;
2. Good localization - the points marked as belonging to an edge should be as close to the center of the edge as possible;
3. Single response - each edge should only be identified once.

In that same paper, Canny proposed his own algorithm for edge detection, which is to date still a very suitable algorithm. Nevertheless an extension was proposed by Deriche [Der87], and several other algorithms have been proposed. Most of these double as corner detectors and will be mentioned later, but one should be noted now: the Sobel operator, based around the notion of convolving the image to acquire an approximation of the derivative, is a filter designed to



Figure 2.1: An example of edge detection [McL10]

emphasize edges in an image [RFW03]. Both the Canny algorithm and the Sobel operator are implemented in OpenCV.

#### 2.1.1.2 Corner detection

Corner detection (see figure 2.2), also called interest point detection, is concerned with finding single, isolated points of interest in an image. These are useful for motion tracking/detection, for 3D modelling and for object recognition. Corners are defined as the intersection of two edges, however most so-called corner detectors actually detect points of interest in general, including isolated points with significant changes in color or brightness, for example.

Moravec defined one of the earliest algorithms designed to detect corners in [Mor80]. This algorithm had limitations pointed out by Moravec himself, and was further improved by Harris and Stephens [HS88]. Their algorithm can also be used to detect edges. It was further improved by Shi and Tomasi in 1994, who managed to modify it so that it shows better results [ST94]. These later two algorithms are implemented in OpenCV.

#### 2.1.1.3 Blob detection

Blob detection (see figure 2.3) focus on the identification of regions of interest in an image. A region of interest is defined as a set of connected points which share some common property such as brightness or color. This property can be constant, or vary within a range of values, but it will always be different from other points surrounding the blob.

Detection is usually either intensity-based or structure based. Intensity-based detection is based around the concept of defining the property being analysed as a function of position and then differentiating this function and looking for local maxima and minima, effectively looking for sudden changes in intensity in the picture. Structure-based detection relies on detecting features such as edges, curves and lines and then using this information to estimate where objects are located.

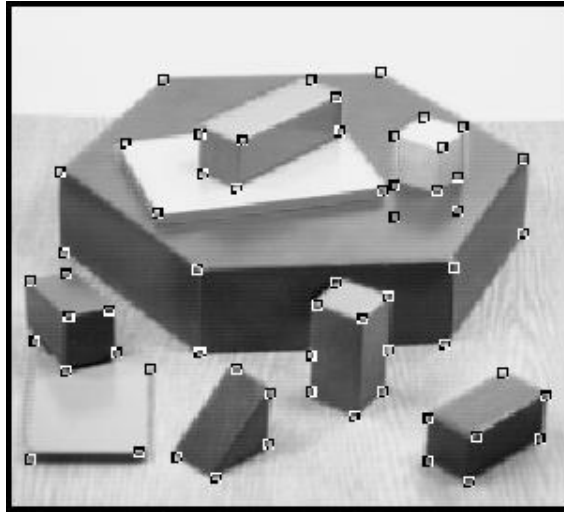


Figure 2.2: An example of corner detection [figa]

Additionally, detectors can also be classified regarding their (in)dependence from scale: they can be fixed scale, scale invariant or affine invariant. These last ones are able of detecting features even when the image is subject to an affine transformation.

A common intensity-based method of blob detection is the Laplace of Gaussian (LoG). A Gaussian filter is first applied to the image. Afterwards, the Laplacian of this filtered image is taken, and blobs are considered to be located where the result takes extreme values (negative values for dark blobs and positive values for bright blobs). This process is analogous to using the Gradient to detect edges, but works for 2D structures [Voo87]. LoG suffers, however, from being sensitive to the size of the kernel used for Gaussian filtering and therefore not being scale invariant. Because of this, it is usually used in a scale-normalized fashion as defined by Lindberg [Lin98].

Lindberg also analysed another way of detecting blobs based around the concept of taking the determinant of the Hessian matrix of the Gaussian-filtered image. Once more, by locating maxima and minima, blobs are identified. [Lin12]

A significantly different approach to blob detection was proposed and used by Matas et al. in [MCUP02] to establish common points between two images taken from different perspectives. Their technique, named "Maximally stable extremal regions" is based around the concept of identifying regions limited by a boundary which remain stable as long as possible when varying a brightness threshold which determines whether a pixel is within a region or not.

All of the aforementioned detectors are intensity-based. However, in 2007 Deng et al. proposed a structure-based detector named "Principal Curvature-Based Region Detector for Object Recognition" [DZM<sup>+</sup>07]. The idea behind PCBR is that while light intensity or color may change and therefore make intensity-based methods inadequate, there are structural elements of objects which can be used to identify them.



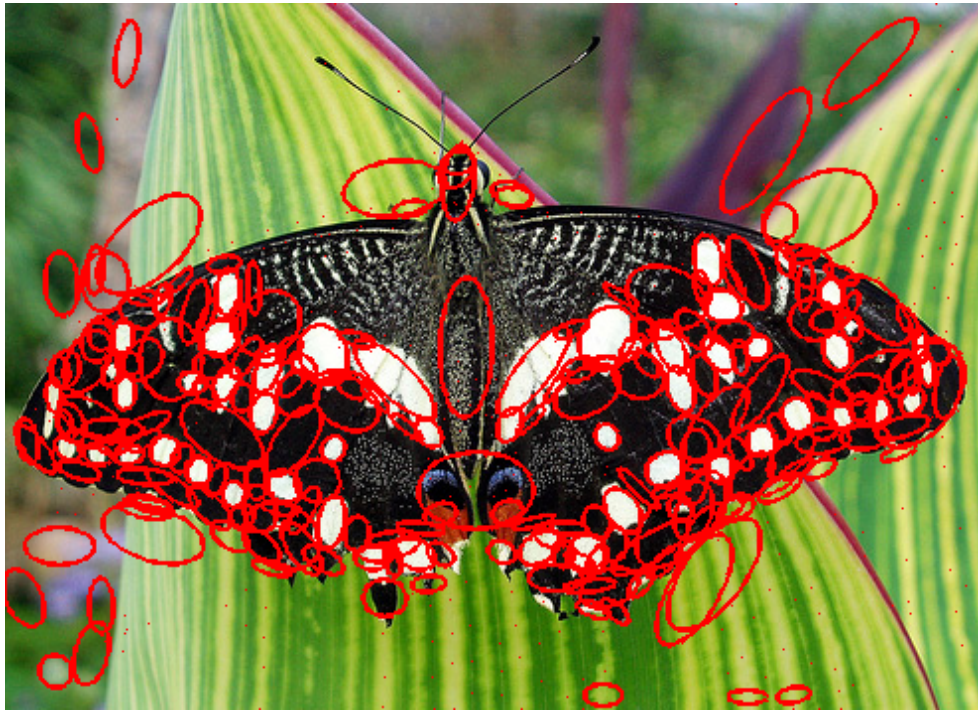


Figure 2.3: An example of blob detection [Guy08]

### 2.1.2 Feature tracking

When dealing with sequences of images (that is, video), it is common that once features have been identified, one wants to track them across multiple frames to detect/describe motion of objects or of the camera, as exemplified in figure 2.4.

A common technique to achieve this is to apply the Lucas-Kanade algorithm [LK81] to estimate optical flow - the perceived motion of an object. This algorithm relies on two basic assumptions [odtb]:

1. A pixel's intensity remains constant between two consecutive frames.
2. Neighbouring pixels move in a similar fashion.

In fact, the Lukas-Kande algorithm takes the 3x3 matrix of pixels centred on the feature being considered and uses that patch for tracking. This algorithm, however, is only applicable if the motion is small enough. If the motion is too fast, it's common to apply it iteratively, starting with low-resolution versions of the image and moving onto higher-resolution improving the model obtained with the coarser images [Hoi]. Further improvements have been achieved by carefully choosing the features to track by applying the Shi-Tomasi feature detector which was mentioned earlier. This forms the basis for the Kanade-Lucas-Tomasi feature tracker (often called KLT).

The algorithms described in the previous paragraph apply to tracking a set of sparse features. However, if one wishes to compute the optical flow of a dense set of points (e.g. all of the pixels in a frame), there is another algorithm, described by Farneback [Far03].

Both Lucas-Kanade and Farneback's algorithms are provided by OpenCV.

A different approach to feature tracking is the usage of a Kalman filter [WB95] to estimate the position of some interesting object. This technique is robust against occlusions (see figure 2.5) and has been used for vehicle tracking with success [SSFD11].



Figure 2.4: An example of feature tracking [figb]

### 2.1.3 Background subtraction

Background subtraction (see figure 2.6) aims at removing background information from an image, assuming all interesting objects are located in the foreground. It is usually applied to video data, and is based around building a model of the background (often initialized from some sub-set of the frames) and then subtracting that model from the frame being processed. It is useful to exclude useless data, and reduce the complexity of further processing steps. As Piccardi put it,

“Background subtraction is a widely used approach for detecting moving objects from static cameras.”

[Pic04]

The most significant step in background subtraction is the modelling of the background, as the quality of the results directly correlates to this. It is desirable that the model adapts to changes in

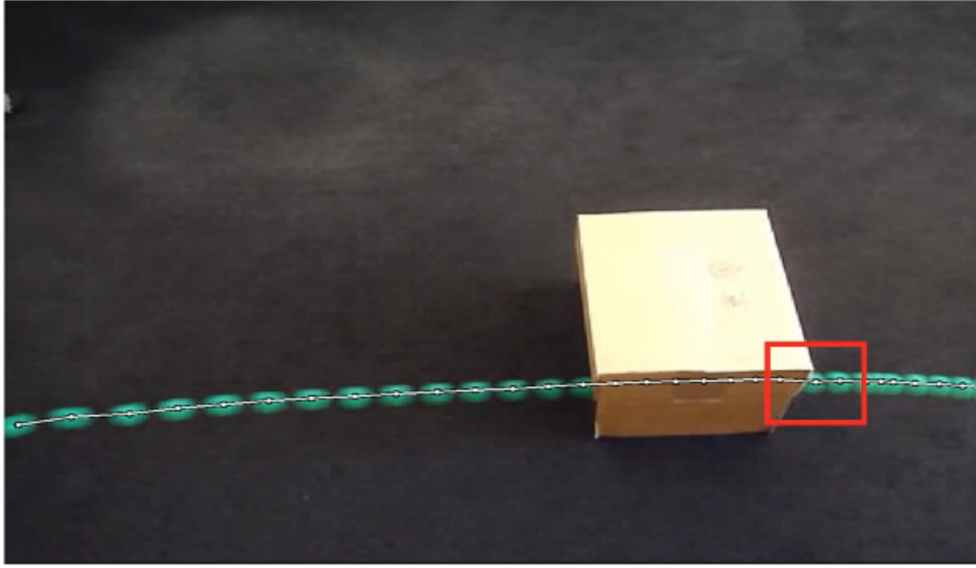


Figure 2.5: Kalman filter being used to track a ball moving from left to right, even when occluded. The point where occlusion ends is marked in red [Kau]

illumination, to new objects entering the scene and remaining there and to the background varying in geometry.

Due to the limited computational resources of the past, it was not always possible to guarantee all of these properties. Therefore, methods have evolved over time. The earliest proposed methods simply modelled a pixel's values over time by fitting a Gaussian probability density function by means of keeping a running average of the pixel's values [WADP97] [KWH<sup>+</sup>94]. Later on, it was proposed that it might be more interesting to instead use the median value of the pixel's last  $n$  values [LV01] [CGPP03].

The problem with both of these approaches, however, is that they do not consider the situation where a pixel might actually be inhabited by different background objects alternately (e.g. when foliage covers a building and moves back and forth due to wind). To mitigate this issue, Stauffer and Grimson presented a new method, by modelling each pixel as a mixture of Gaussians, and deciding which Gaussians represent background colors. Therefore, pixels are considered foreground if their value does not fit into any of the Gaussians which is considered as modelling a background color for that pixel [SG99].

The original Mixture of Gaussians as proposed by Grimson and Stauffer has been improved and modified over the years. The one proposed in [KB01] improves upon the original's learning speed and accuracy and introduces shadow differentiation. This algorithm is known as MOG in OpenCV. Another variation, proposed by Zivkovic in [Ziv04], introduces the notion of varying the number of components for the mixture of Gaussians between pixels and over time, and therefore makes the algorithm more adaptable to changing environments. This algorithm is implemented in OpenCV as MOG2 [odta].



Figure 2.6: An example of background subtraction. In the lower half, white represents foreground and black represents background [figc]

A different approach is the one proposed by Godbehere et al. in [GMG12] which is based around inferring which pixels belong to the foreground based on an Histogram of previous pixel values. The algorithm described in the paper also preforms feature tracking, but some implementations are limited to the Background Subtraction portion.

#### 2.1.4 Morphological operations

Morphological operations are binary operations on an image which are related to the shape (morphology) of the different structures in the image. In the context of Computer Vision they are normally used to remove noise and make detected features more prominent [Eff00].

They consist in placing a structuring element over all possible positions in an image, and applying some binary operation to the covered pixels based on their values as compared to those in the structuring element. The structuring element can be square, but can also be diamond-shape, cross-shaped, circular, or a myriad of other formats when interesting [Del].

The basic operations are erosion and dilation. Erosion consist in setting all pixels under the area covered by the structuring element to 0 when at least one of them isn't 1 - effectively removing a layer of outer pixels from all regions in the image. Dilation does the opposite, setting all pixels covered by the structuring element to 1 if at least one of them is 1, and thus adding a layer of pixels.

Erosion and dilation can further be combined to form two other interesting operations: opening and closing. Opening is an erosion followed by a dilation and takes its name from the fact that the resulting image will have any thin connections between elements opened into gaps. Closing is the opposite - a dilation followed by an erosion - and derives its name from closing any internal gaps in elements while keeping the outer size the same.

### **2.1.5 Overview**

There are several areas of Computer Vision, each with different goals. This section looked into those which are considered to be related with the task at hand. Regarding the typical flow of a CV algorithm, generally speaking, one should first apply some sort of information filtering (background subtraction or feature detection), and then process the resulting data to find interesting data about the images being analysed (feature tracking, statistical treatment, etc.).

## **2.2 Traffic monitoring**

Aside from existing Computer Vision techniques, explored in the previous section, it is also important to take a look at existing road traffic monitoring solutions, as well as the associated models.

### **2.2.1 Road traffic monitoring solutions**

Historically, there have been two very distinct ways of detecting road traffic: physical sensors placed on/near the road which detect the presence of a vehicle such as loop sensors, microwave detectors and pressure tubes, and computer vision solutions which apply computer vision algorithms to images captured by a camera aimed at the road and detect and count vehicles [Set97].

As established by Setchell, physical sensors provide useful data but are costly and cumbersome to install and maintain, which partially explains the motivation and advent of computer vision based tracking methods. These, in turn, can be divided into two broad categories: those based on fixed cameras and those based on cameras installed in UAVs.

#### **2.2.1.1 Fixed camera**

Fixed camera solutions consist in one or more cameras installed in, for example, a light pole or a road sign, in such a way that they can observe a considerable portion of the road. They're installed at an angle and usually between 5m and 10m high. They may be able of panning, tilting and zooming, but it is not required.

It is worth nothing that the systems being discussed here are based on electronic cameras. There is previous work done with analogue equipment, such as that done by Greenshields in the 1930s [Kü08] but given that the focus of this current work is to apply computer vision algorithms, this will not be considered.

These systems were first explored in [TSK<sup>+</sup>82] and then further reviewed by Iñigo in [Ini85] and [Ini89]. In his first paper, Iñigo noted that, at the time, there were no computer vision based

road traffic monitoring systems in place except in Japan, largely because of the lack of computational power and the high cost of cameras at the time. He predicted, however, that these costs would drop and make these solutions feasible. These predictions were reiterated in his second paper, where a CV based system is described, which detected vehicles by detecting changes in illumination along a line perpendicular to a lane being monitored. That same paper also mentions one of the major disadvantages of fixed camera systems: the obligatory presence of occlusions, because the camera is installed at an angle relative to the road. Additionally these systems also suffer from being relatively inflexible in the sense that they can not be easily moved and therefore only observe one specific stretch of road.

Later on, Koller et al. tackled the problem of Occlusion in [KWM94] by introducing a “Occlusion Reasoning” step into the vehicle tracking algorithm. Coifman et al., on the other hand, decided to track features instead of whole vehicles in an attempt to mitigate the same issue [CBMM98]. Rad and Jamzad proposed a solution which tracks vehicles by establishing bounding boxes and predicting where they might be next based on the size of the bounding box and a fixed. They also mitigate occlusion by tracking the centres of the bounding boxes and predicting occlusion when they come too close together [RJ05].

More recently, Hsieh et al. have proposed a novel system which detects lane divisions and then uses that data to more easily distinguish between vehicles. Additionally, their system is also able to distinguish between different types of vehicles [HYCH06]. Also recent is some work which is based upon the idea of taking video streams from the internet and using those as data sources [LRB09].

#### 2.2.1.2 UAV installed camera

An UAV (unmanned air vehicle) is an aircraft which, rather than having a pilot on board, is either controlled remotely or operates autonomously. These types of vehicles have been commonly referred to as “drones”. Although UAVs are commonly associated with military operations, and even though they were first developed with that purpose in mind [Sha], nowadays there are several civil and scientific applications for this kind of vehicle. One such is traffic monitoring, as surveyed by Kanistras et al. in [KMRV13]. Even though some of the solutions mentioned do not rely on video data, only those which do will be considered, given that the others are out of the scope of the current work.

When it comes to acquiring and analysing video data with an UAV, two major dimensions can be considered: whether the UAV moves or not while capturing data, and whether this data is analysed by an on-board computer, or transmitted to be analysed remotely.

The question of movement is important because if the UAV moves, the background will certainly change drastically as the capturing of data goes on, and therefore background subtraction techniques are most likely not appropriate. In these cases, one will have to resort to tracking interesting features, and keeping track of the camera’s position relative to the road is almost a necessity. An interesting algorithm which deals with detecting road features and limits is proposed



in [Kim05]. On the other hand, if an UAV remains stationary over a section of road, background subtraction techniques similar to those used for stationary cameras might be of interest.

As far as data transmission goes, the question which must be asked is whether the available network provides enough throughput to transmit video data in real-time. Even if it does, there might be interest in having the actual UAV doing the processing itself, if one wishes it to react to changing traffic conditions (e.g. follow a traffic jam to locate the cause). If one opts to do the processing on the UAV itself, then the computational resources will most likely not be as plentiful as those available on the ground, so care must be taken to ensure the algorithm can run in real-time.

## **2.2.2 Models**

Once data about how vehicles move is acquired, it is important to know how to use this data to model a section of road. There is some work regarding what these models should describe and how they can be generated.

### **2.2.2.1 Fundamental diagram of traffic flow**

The fundamental diagram of traffic flow is a diagram first used by Greenshields [Kü08] to show the relationship between vehicle Speed, Flow and Density in a stretch of road. Flow is given as vehicles/hour, Density as vehicles/kilometre and Speed as kilometre/hour. Greenshields postulated that  $\text{Flow} = \text{Density} * \text{Speed}$ .

#### **2.2.2.2 Statistical traffic models**

As far as the generation of statistical models goes, the interesting variables are the aforementioned flow, density and speed, as well as, for intersections, mapping of entrances to exits in terms of numbers of vehicles. There has been some work regarding the generation of these models by Puri et al. [PVK07] and by Coifman et al. [CMM<sup>+</sup>06].

State of the art



## Chapter 3

# Solution

This chapter presents the algorithm which was designed to tackle the problem at hand, as well as the process which led to the design choices which were made. Afterwards, the constraints that it imposes on the input data will be presented, as well as the implementation which was used for tests.

### 3.1 Overview

The general approach to the problem was that vehicles had to first be identified, to then be tracked. The result of the tracking process is a set of trajectories which are then used to build a model of the intersection.

The identification of vehicles is done with background subtraction. However, due to camera sway, sudden movements had to be detected and compensated against. Once that is done, blobs which are classified as foreground are considered vehicles and tracked from frame to frame.

The factors which led to choosing this approach, as well as the development process are detailed in section [3.2](#).

### 3.2 Development process

This section describes the algorithm which was developed, along with the process which led to the choices which were made regarding which algorithms to use. The flowchart of the algorithm is shown in figure [3.1](#).

The first step in developing the algorithm was to find a way to identify vehicles. The most common technique for this which was identified in previous research is background subtraction, so the choice was made to experiment with MOG, more specifically the MOG2 variant as implemented by OpenCV [[Ziv04](#)], due to its popularity and it being the most recent variation of MOG in that library. With the camera in a stable position this proved to work correctly. However, it was quickly discovered that camera instability led to MOG failing to correctly identify foreground and background. In extreme cases, the whole frame would be considered foreground, as all of the

## Solution

pixels had moved too far from their original position and none matched the statistical model kept by MOG. This led to the necessity of finding some stabilization technique.

The first attempt was to find some fixed features (such as rooftop details) in the image to be tracked from frame to frame, and to be used to infer a transformation matrix to be applied to the image, so that the image was distorted in order to bring objects back to their relative position before the camera shake. However, this proved unsuccessful, as there were no sufficiently unique features in the image to be tracked from frame to frame when the camera moved suddenly, and therefore the transformation matrix was wrongly inferred. This was observed empirically, by running the algorithm and looking at the transformed frames, which sometimes were grossly distorted, to the point where they were useless, as can be seen in figure [3.2](#).

The second attempt was to calculate the optical flow of the image, to, again, infer a transformation matrix. This too was unsuccessful, as the high amount of rapidly moving vehicles would introduce significant error into the calculation of optical flow.

The solution which, finally, was successful was to identify camera sway by monitoring the ratio of foreground pixels identified by MOG from frame to frame, and considering sway to be occurring when a sudden increase in this ratio was detected (via trial and error, a ratio threshold of 1.05 was found to work well). It should be noted that this increase depends on frame rate, and thus for different frame rates, the ideal value might be something other than the one which was identified. Once sway is identified, the algorithm then waits until the camera stabilizes (that is, the ratio drops below the threshold), and then interpolates between the last stable before instability occurred and the first stable frame following that section.

This choice proved to work well, but led to the choice of making the algorithm work in two phases, as it is first necessary to process the whole of the video to identify sections of instability, interpolate them and identify vehicles. Only once this is done, can the algorithm then move onto identifying the paths the vehicles take. This choice also facilitated the development process, given that it made it possible to first perfect the interpolation phase, and only afterwards move onto vehicle identification and path building.

As far as vehicle identification, the common technique in Computer Vision was used - once the foreground is identified, independent blobs are identified by finding their contours. Empirically, it was found that first running one erosion and two dilations on the frame improved results, as it removed noise and connected vehicles which might be split into two accidentally. This operation lead to, sometimes, two separate vehicles being identified as one single block. That problem is addressed further down.

Unfortunately, the result of the previous operation still proved to have a lot of false positives, so a filter had to be devised. Initially, the attempt was made to restrict the bounding box's aspect ratio, but unfortunately this caused too many false negatives, as the bounding box would change ratio when vehicles turned, sometimes to values very close to those of the noise blobs' bounding boxes. The choice was made to impose a restriction on the area of the blob's bounding box. Because the area of the bounding box will change with the input resolution and the level of zoom

of the camera, this value had to be modifiable by the user, so it was included in the configuration file. The result of blob filtering is shown in figure 3.3.

Once vehicles are identified, it is then necessary to track them from frame to frame. Because the sections of instability are interpolated, sometimes vehicles will appear to "freeze" and then jump forward. Additionally, sometimes vehicles will be partially occluded, for example, by cables hanging above the intersection. To mitigate these issues, a Kalman filter is used for each vehicle to predict its position in the following frame. This filter is initialized with the vehicle's position when it first identified and the position of the vehicle closest to it (by Cartesian distance) in the next frame. Afterwards, the Kalman filter is used to predict the position of the vehicle in a following frame. If a vehicle is found sufficiently close to the prediction, it is considered the next vehicle. If it is not, then the prediction is used as the next position and the process continues. However, if no suitable match is found after 5 usages of the subsequent predictions, the trajectory is considered to have ended. This process prevents "losing" vehicles both due to interpolation of frames and due to a vehicle not being detected in a certain frame for whatever reason, but also, due to the limit of subsequent predictions, prevents the algorithm from artificially continuing a trajectory once the vehicle has effectively disappeared.

This process works when vehicles remain roughly on the same trajectory during periods of camera instability. Additionally, this process is also robust against vehicle merges due to proximity, as it keeps a Kalman filter of each vehicle originally identified, and therefore when the vehicles separate each trajectory will be distinguished once again.

After all frames have been processed, only trajectories which intercept at least two limits are considered as being interesting. This eliminates cases where some noise at the edge of the frame or the middle of the intersection was considered to be a vehicle moving erratically. The limit they first intercept is considered the "entrance" they take, and the limit they intercept second is considered the "exit".

Finally, trajectories which differ from one another in less than 5% of their points are condensed into one trajectory. This is done because, sometimes, long vehicles are detected as multiple blobs, due to being partially occluded by cables or moving too fast, and are then tracked as multiple vehicles even if the blobs merge later on. This causes false positives which skew the count. The choice of 5% was based on the fact that these divisions occur, normally, at the entrance of the intersection, which accounts for roughly that amount in a vehicles trajectory.

The algorithm is strong against false positives thanks to the combination of ignoring blobs with nonsensical aspect ratios, as well as ignoring trajectories which do not both enter and exit the intersection or are clear duplicates of others. Ignoring trajectories which do not intersect an entrance, however, means that it does not consider any vehicles which were already inside the intersection when the video started. Therefore the algorithm requires a warm up period. On the other hand, this algorithm may ignore too much and therefore have an high ratio of false negatives, specifically in what concerns bicycles and motor bikes. This drawback could be minimized by improving the heuristic which filters the blobs to be tracked.

### 3.3 Technical details

#### 3.3.1 Input data

The algorithm takes, as input, a video file to be analysed, a configuration file containing the list of lines which delimit the intersection and the expected minimum area in pixels of each vehicle, the number of frames to be skipped at the beginning of execution, the number of frames to be used for the learning phase of MOG, and, optionally, the maximum number of frames to process.

#### 3.3.2 Output data

The algorithm returns, as output, a matrix of  $X$  by  $X$  (where  $X$  is the number of lines delimiting the intersection), containing the mapping of number of vehicles that enter the intersection at a certain limit, and exit at a certain limit.

#### 3.3.3 Configuration file format

Because there are parameters which depend on the input video, such as the intersection's limits, which are identified manually, it is necessary to feed this information into the algorithm. The choice was made to provide this information via a configuration file, so that these parameters can be stored permanently alongside the video, and to reduce the number of command line arguments.

The configuration file is a plain-text file used to identify the limits of the intersection. It has a single number  $S$  on the first line, corresponding to the expected minimum area of a vehicle in pixels. It has a single number  $N$  on the second line, corresponding to the amount of intersection limits which will be listed, followed by  $N$  lines, specifying one limit each. A limit specification has four different numbers separated by spaces, corresponding to the starting  $x$ , starting  $y$ , ending  $x$  and ending  $y$  coordinates, respectively. These coordinates are specified in pixels.

#### 3.3.4 Algorithm parameters

The solution uses several computer vision algorithms which take configuration parameters. These are detailed below.

##### 3.3.4.1 MOG

MOG was configured with the following values:

- **history:** 1 - this value was chosen after preliminary testing suggested this worked well;
- **varThreshold:** 16 - this is the suggested value on the openCV documentation pages;
- **bShadowDetection:** false - there is no interest in detecting shadows;
- **learning rate:** 0.003 - once again, chosen due to preliminary testing suggesting this was a good value.

### 3.3.4.2 Kalman filter

The Kalman filters for each vehicle were all configured with the same values:

- **Dynamic parameters:** 4, corresponding to position and velocity in the x and y coordinates;
- **Measure parameters:** 2, corresponding to position and velocity in the x and y coordinates;
- **Transition matrix:**

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Meaning that

$$x_{t+1} = x_t + v_{x_t}$$

$$y_{t+1} = y_t + v_{y_t}$$

$$v_{x_{t+1}} = v_{x_t}$$

$$v_{y_{t+1}} = v_{y_t}$$

That is, it is assumed the velocity of vehicles traversing the intersection remains constant.

- **Process noise covariance matrix:**  $4 \times 4$  matrix with all values equal to  $1 \times 10^4$  - empirically found to be the lowest value which resulted in accurate predictions. Lower orders of magnitude would result in erratic predictions;
- **Measurement noise covariance matrix:**  $4 \times 4$  matrix with all values equal to 0, given that measurements are assumed to be exact;
- **Posteriori error estimate covariance matrix:**  $4 \times 4$  matrix with all values equal to 0.

### 3.3.4.3 Other parameters

Aside from the parameters of other algorithms which are used, the solution itself also has some parameters namely:

- **Minimum area of blob bounding box:** this parameter is alterable by the user, as it depends on the specific clip being analysed.
- **Maximum distance between two subsequent positions of a vehicle:** this value is used to prevent the algorithm from jumping between two unrelated vehicles. 30 pixels was originally found to be a good value for the clip which was used for testing, and was kept as it worked well in all performed tests.
- **Maximum number of subsequent predictions:** Meaning the maximum amount of subsequent times the Kalman filter prediction can be used instead of an actual measurement.

5 was chosen because it was found that vehicles usually were not identified at most for 2 subsequent frames, and interpolations were usually, at most, 4 frames long.

### 3.4 Limitations

Due to the technique used to mitigate camera shake and the hard coding of the intersection limits, the camera must remain relatively static for duration of the video. Any permanent shift in perspective leads to the algorithm not working. Any prolonged shake will cause too many frames to be missed and the tracking phase of the algorithm to not be able to follow the vehicles.

### 3.5 Implementation

The algorithm was implemented in C++, using the OpenCV library to represent and manipulate video images, as well as for its implementation of several computer vision algorithms which are used - MOG2, contour identification, minimum area rectangle identification and Kalman filters. The complete implementation can be found in

<http://paginas.fe.up.pt/~ei10107/VehicleTrack.rar>.

Additionally, the implementation has a GUI [3.4](#) which allows the user to navigate the results of the execution back and forward, inspecting identified vehicles and trajectories.

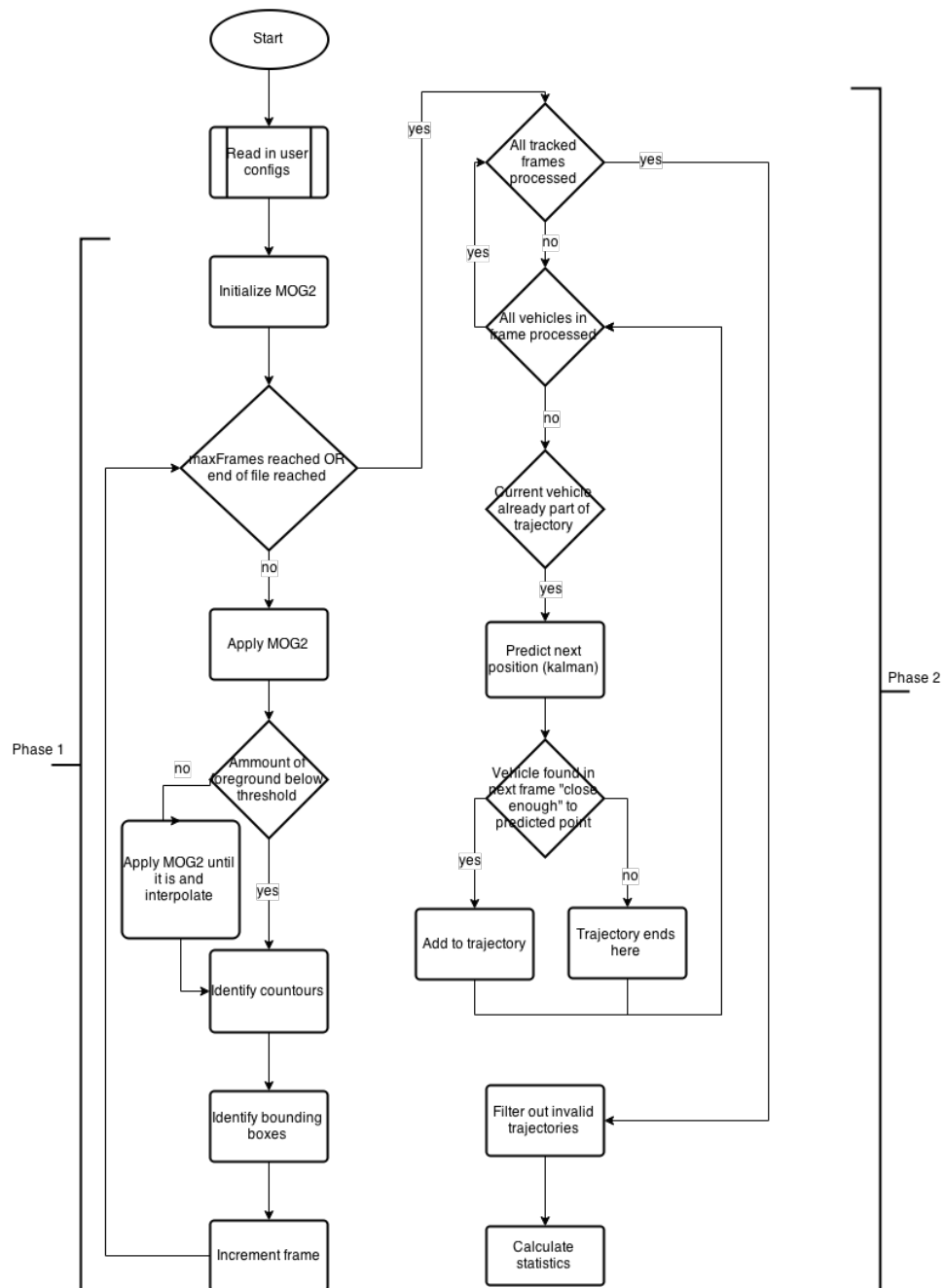


Figure 3.1: Flowchart of the general execution of the algorithm



## Solution

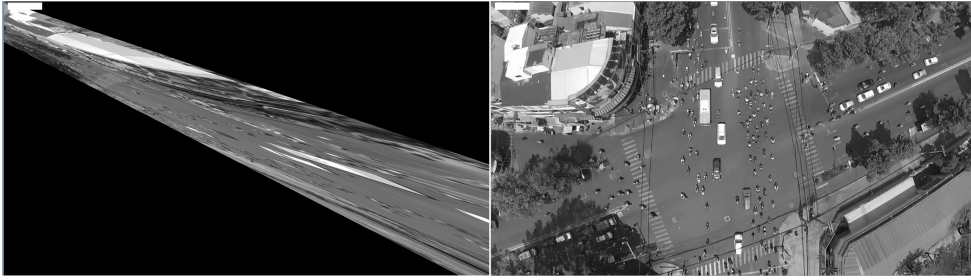


Figure 3.2: Distortion resulting from attempting to find a transformation matrix (left) and reference frame (right)



Figure 3.3: Same frame before (top) and after (bottom) blob filtering. Note the disappearance of the small blobs identified in the trees, zebra crossings and hanging wires.



## Solution

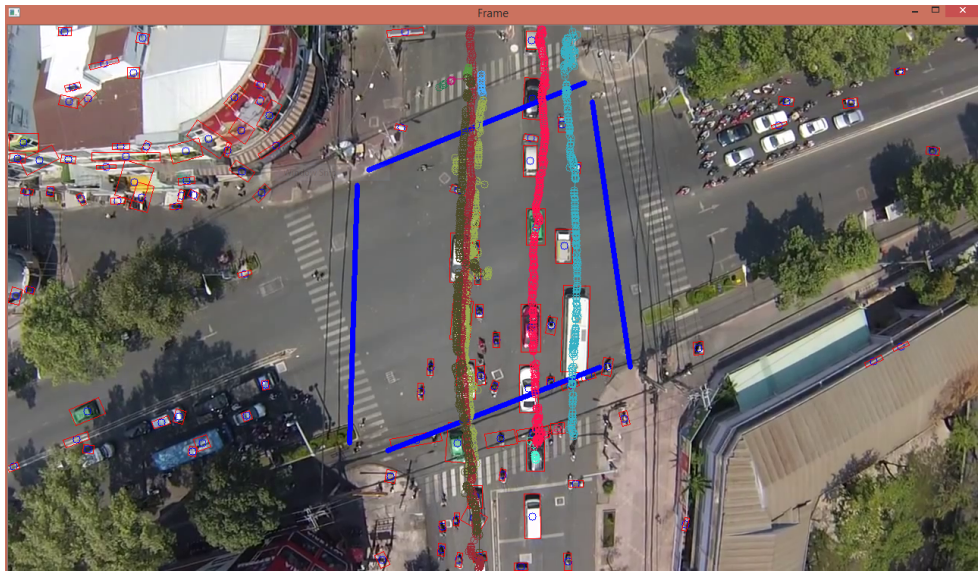


Figure 3.4: Example of the GUI being used to show each point in several trajectories

Solution

## Chapter 4

# Experimental results

This chapter describes the process which was used to assess the solution described previously. The success metrics which are used will be considered first, followed by a description of the data set which was fed to the algorithm, and then the corresponding results and their analysis.

### 4.1 Metrics

One interesting metric for success is the percent of vehicles correctly identified and tracked - the amount of true positives. On the other hand, it is also important to measure the amount of false positives and false negatives as these can skew the statistical model. The analysis of false positives requires looking at the algorithm in action using the GUI which was developed, in order to check which of trajectories being identified don't actually match a true vehicle.

The relationship between true positives ( $TP$ ), false positives( $FP$ ) and false negatives( $FN$ ) is given by the following two formulas:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

That is, precision measures how many items are accurate, of those identified, while recall measures how many items are identified, of those which should have been.

Another interesting metric is the distribution of traffic per pair of entrance/exit, based on percentage of vehicles in that pair per total amount of vehicles counted. This distribution is represented as a square matrix. Two of these distributions can be compared by using the matrix analogous of the Euclidean norm for vectors, given by the following formula:

$$d = \frac{\sqrt{tr((M - G)^T \times (M - G))}}{n_{entrances}}$$

where  $M$  is the measured distribution,  $G$  is the ground-truth distribution and  $tr$  is the trace of the resulting matrix. This results in  $d$  being the square root of the sum of the squares of the differences

of each element of the matrices, divided by the number of entrances of the intersection. This division is necessary because different intersections have different numbers of entrances and it is desirable to normalize the value of  $d$  to compare between such cases.

The final metric is execution time - how fast the algorithm the input data is very important for real world applications. This depends both on the resolution of the input data and its complexity, as scenes with more camera shake or more vehicles are harder to process.

### 4.2 Setup and test cases

In order to test the algorithm, it would have been ideal to have access to multiple video clips with the desired characteristics (vertical perspective, high stability and considerable length). However, no clips within those parameters could be found beyond the one which was originally identified. To mitigate this issue, some footage was captured manually, emulating the perspective which would result from using an UAV.

Thus, there are two clip sources:

1. **Source 1** A UAV-capture clip of a 4-way intersection in Vietnam which was extracted from an online source, and used to build the algorithm, as well as for testing
2. **Source 2** A manually captured clip of a 3-way intersection in Porto, which was captured manually, and used exclusively for testing

Each source provided several clips all of which were processed in three different resolutions, all corresponding to a 16:9 aspect ratio:

- 720p - At 1280 by 720 pixels, commonly referred to as "HD";
- 480p - At 854 by 480 pixels;
- 360p - At 640 by 360 pixels;

In all cases, the algorithm was ran using the 250 frames preceding the clip to teach MOG2. All tests were preformed on the same machine - a PC running Microsoft Windows 8 64bit, with an Intel i7 4710HQ (2.50 GHz) processor, 12Gb of memory and reading the data from an HDD.

#### 4.2.1 Source 1

Source 1 shows traffic flowing in a 4-way intersection, with 4 limits, identified in figure 4.1. In this sub-section, traffic between 2 and 3 (and vice-versa) will be referred to as "vertical traffic" and traffic between 1 and 4 (and vice-versa) will be referred to as "horizontal traffic".

In order to have some diversity in the tests, this source was used in three different fashions:

1. **Clip 1** - The entirety of the footage, which consists of essentially of 1.5 traffic light cycles. That is, traffic flows vertically, then horizontally, and then vertically once again. The clip is 3 minutes and 10 seconds long, corresponding to 4750 frames;



Figure 4.1: Identification of the source 1 intersection limits. 1 - left; 2 - bottom; 3 - top; 4 - right

2. **Clip 2** - 1000 frames, corresponding to the first 40 seconds, at the beginning of the clip, which consist of the first cycle of vertical traffic;
3. **Clip 3** - 1000 frames, corresponding to the last 40 seconds, at the end of the clip, which consist of the second cycle of vertical traffic;

The period with the most horizontal traffic, roughly corresponding to the middle 2.5 minutes coincides with a period of high camera instability, and, at the very end, a very sharp zoom out, which means it is not as interesting to analyse as the other sections. This is why no partial clip was extracted from the middle of the original footage.

#### 4.2.2 Source 2

Source 2 shows traffic flowing in a 3-way intersection which is highly irregular. Traffic flows uncontrolled by traffic lights, therefore vehicles cross paths often. Figure 4.2 illustrates the entrances/exits in this intersection. Two clips resulted from this source:

1. **Clip 4** - 1 minute and 15 seconds of low density traffic, distributed nearly uniformly;
2. **Clip 5** - 1 minute and 11 seconds of medium density traffic with most vehicles travelling from entrance 1 to entrance 3

### 4.3 Characterization of the clips

Table 4.1 shows the number of trucks, cars and buses passing through the intersection from source 1 during the duration of each clip. Each table has 4 columns and 4 rows, representing each limit

## Experimental results

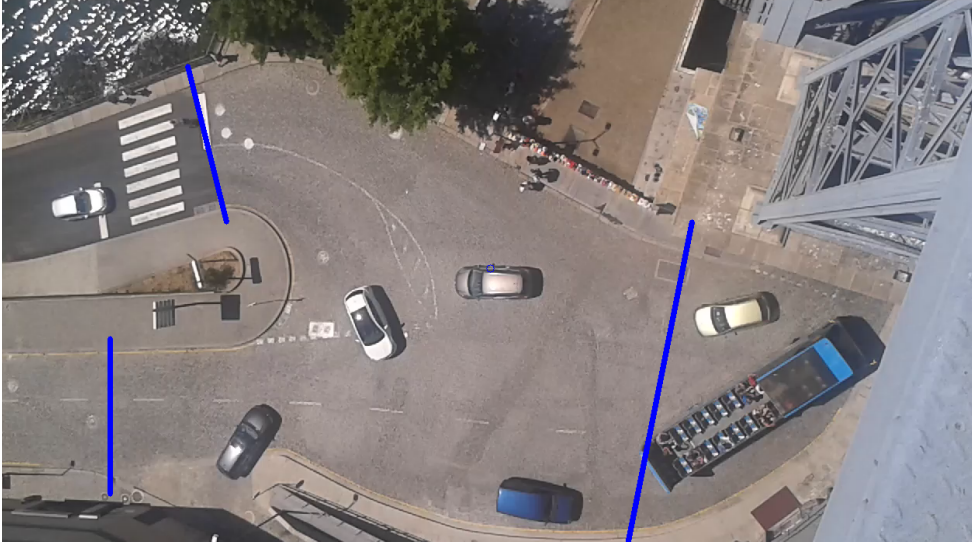


Figure 4.2: Identification of the source 2 intersection limits. 1 - top left; 2 - bottom left; 3 - right

of the intersection. The number on cell (C,R) where C is the column number and R is the row number, represents the number of vehicles going from entrance C to exit R.

Table 4.5 shows the distribution of traffic based on the percentage of vehicles (out of the total for that clip) going from a certain entrance to a certain exit.

Table 4.1: Ground-truth values. Columns represents entrances and rows represent exits

Table 4.2: Clip 1					Table 4.3: Clip 2					Table 4.4: Clip 3				
	1	2	3	4		1	2	3	4		1	2	3	4
1	0	0	5	10	1	0	0	1	0	1	0	0	3	0
2	3	0	23	0	2	0	0	8	0	2	0	0	8	0
3	0	25	0	0	3	0	13	0	0	3	0	5	0	0
4	21	4	0	0	4	0	0	0	0	4	0	1	0	0

Table 4.5: Distribution of traffic given the ground-truth values

Table 4.6: Clip 1					Table 4.7: Clip 2					Table 4.8: Clip 3				
	1	2	3	4		1	2	3	4		1	2	3	4
1			5%	11%	1			5%		1			18%	
2	3%		25%		2			36%		2			47%	
3		27%			3		59%			3		29%		
4	23%	4%			4					4		6%		

The number of motorcycles is not shown because, unfortunately, the detection rate on motorcycles is close to null. There are a series of reasons for this which will be explored at the end of the current chapter.

## Experimental results

Table 4.9 shows the number of vehicles passing through the intersection during the duration of each clip from source 2. Each table has 3 columns and 3 rows, representing each limit of the intersection. The columns and rows have the same meaning as before

Table 4.12 shows the distribution of traffic based on the percentage of vehicles (out of the total for that clip) going from a certain entrance to a certain exit.

Table 4.9: Ground-truth values. Columns represents entrances and rows represent exits

Table 4.10: Clip 4

	1	2	3	4
1	0	3	2	
2	0	0	1	
3	3	2	0	

Table 4.11: Clip 5

	1	2	3	4
1	0	0	2	
2	3	1	0	
3	10	1	0	

Table 4.12: Distribution of traffic given the ground-truth values

Table 4.13: Clip 4

	1	2	3	4
1		27%	18%	
2			9%	
3	27%	18%		

Table 4.14: Clip 5

	1	2	3	4
1			12%	
2	18%	6%		
3	59%	6%		

This data in all of the aforementioned tables is used as the ground-truth in further analysis.

## 4.4 Results

This section shows the counts obtained by the algorithm for the 5 clips in each of the 3 resolutions. It also shows the amount of true positives and false positives/negatives, as well as the precision and recall measurements.

Additionally, it presents the traffic distributions, and respective comparisons to their analogous in the ground-truth.

The experimental results for source 1 showed very poor detection rate for motorcycles (close to or actually 0%, depending on the clip and resolution). Therefore, the results which follow are mostly based on the detection of cars, trucks and buses. However, because some motorcycles were in fact detected, they need to be accounted for, otherwise the number of false positives would be artificially inflated.

On the other hand, for source 2 motorcycles did not prove to be a challenge, therefore they are not considered a special case and are simply counted normally.

### 4.4.1 Clip 1

The results for clip 1 are in tables 4.15, 4.19, 4.23 and 4.27.

## Experimental results

Table 4.15: Results for clip 1 in 360p

Table 4.16: Vehicle count

	1	2	3	4
1	0	1	0	4
2	2	0	4	0
3	1	7	0	3
4	5	1	0	0

Table 4.17: Vehicle distribution

	1	2	3	4
1		4%		14%
2	7%		14%	
3	4%	25%		11%
4	18%	4%		

Table 4.18: Measurements for clip 1 in 360p

<b>True positives</b>	23
<b>False negatives</b>	68
<b>False positives</b>	3
<b>Motorcycles</b>	2
<b>Precision</b>	88.5%
<b>Recall</b>	25.3%
<b>d</b>	0.0468

Table 4.19: Results for clip 1 in 480p

Table 4.20: Vehicle count

	1	2	3	4
1	0	1	1	5
2	1	0	7	2
3	0	13	0	2
4	8	2	0	0

Table 4.21: Vehicle distribution

	1	2	3	4
1		2%	2%	12%
2	2%		17%	5%
3		31%		5%
4	19%	5%		

Table 4.22: Measurements for clip 1 in 480p

<b>True positives</b>	37
<b>False negatives</b>	54
<b>False positives</b>	2
<b>Motorcycles</b>	3
<b>Precision</b>	94.9%
<b>Recall</b>	40.7%
<b>d</b>	0.0321

Table 4.23: Results for clip 1 in 720p

Table 4.24: Vehicle count

	1	2	3	4
1	0	4	2	15
2	0	0	9	4
3	3	18	0	4
4	8	2	3	0

Table 4.25: Vehicle distribution

	1	2	3	4
1		6%	3%	21%
2			13%	6%
3	4%	25%		6%
4	11%	3%	4%	

Table 4.26: Measurements for clip 1 in 720p

<b>True positives</b>	49
<b>False negatives</b>	42
<b>False positives</b>	12
<b>Motorcycles</b>	11
<b>Precision</b>	80.3%
<b>Recall</b>	53.8%
<b>d</b>	0.0591

Table 4.27: Running times for clip 1

Resolution	Time (s)
360p	72
480p	122
720p	302

Overall, the accuracy increases with the resolution, but so does the processing time, as expected. The low accuracy for all three resolutions will be explained further on.



## Experimental results

### 4.4.2 Clip 2

The results for clip 2 are in tables 4.28, 4.32, 4.36 and 4.40.

Table 4.28: Results for clip 2 in 360p

Table 4.29: Vehicle count

	1	2	3	4
1	0	0	0	0
2	0	0	4	0
3	0	6	0	0
4	0	0	0	0

Table 4.30: Vehicle distribution

	1	2	3	4
1				
2			40%	
3		60%		
4				

Table 4.31: Measurements for clip 2 in 360p

<b>True positives</b>	10
<b>False negatives</b>	12
<b>False positives</b>	0
<b>Motorcycles</b>	0
<b>Precision</b>	100%
<b>Recall</b>	45.5%
<b>d</b>	0.0148

Table 4.32: Results for clip 2 in 480p

Table 4.33: 1

	1	2	3	4
1	0	1	0	0
2	1	0	4	0
3	0	6	0	0
4	0	2	0	0

Table 4.34: Vehicle distribution

	1	2	3	4
1		7%		
2	7%		29%	
3		43%		
4		14%		

Table 4.35: Measurements for clip 2 in 480p

<b>True positives</b>	10
<b>False negatives</b>	12
<b>False positives</b>	2
<b>Motorcycles</b>	2
<b>Precision</b>	83.3%
<b>Recall</b>	45.5%
<b>d</b>	0.0638

Table 4.36: Results for clip 2 in 720p

Table 4.37: Vehicle count

	1	2	3	4
1	0	4	1	1
2	2	0	7	0
3	0	13	0	0
4	0	1	1	0

Table 4.38: Vehicle distribution

	1	2	3	4
1		13%	3%	3%
2	7%		23%	
3		43%		
4		3%	3%	

Table 4.39: Measurements for clip 2 in 720p

<b>True positives</b>	21
<b>False negatives</b>	1
<b>False positives</b>	5
<b>Motorcycles</b>	4
<b>Precision</b>	80.8%
<b>Recall</b>	95.5%
<b>d</b>	0.0650

The results for this shorter clip are, overall, better than those for the first clip, and particularly in the case of the 720p version, much better. Again, this will be explored in the analysis section.

### 4.4.3 Clip 3

The results for clip 3 are in tables 4.41, 4.45, 4.49 and 4.53.

## Experimental results

Table 4.40: Running times for the clip 2

Resolution	Time (s)
360p	20
480p	33
720p	75

Table 4.41: Results for clip 3 in 360p

Table 4.42: Vehicle count

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	1	0	0
4	0	0	0	0

Table 4.43: Vehicle distribution

	1	2	3	4
1				
2				
3		100%		
4				

Table 4.44: Measurements for clip 3 in 360p

<b>True positives</b>	1
<b>False negatives</b>	16
<b>False positives</b>	0
<b>Motorcycles</b>	0
<b>Precision</b>	100%
<b>Recall</b>	5.9%
<b>d</b>	0.2171

Table 4.45: Results for clip 3 in 480p

Table 4.46: Vehicle count

	1	2	3	4
1	0	0	0	0
2	0	0	1	0
3	0	2	0	0
4	0	0	0	0

Table 4.47: Vehicle distribution

	1	2	3	4
1				
2			33%	
3		67%		
4				

Table 4.48: Measurements for clip 3 in 480p

<b>True positives</b>	3
<b>False negatives</b>	14
<b>False positives</b>	0
<b>Motorcycles</b>	0
<b>Precision</b>	100%
<b>Recall</b>	17.6%
<b>d</b>	0.1096

Table 4.49: Results for clip 3 in 720p

Table 4.50: Vehicle count

	1	2	3	4
1	0	0	1	3
2	0	0	8	1
3	1	7	0	1
4	1	1	1	0

Table 4.51: Vehicle distribution

	1	2	3	4
1			4%	12%
2			32%	4%
3	4%	28%		4%
4	4%	4%	4%	

Table 4.52: Measurements for clip 3 in 720p

<b>True positives</b>	15
<b>False negatives</b>	2
<b>False positives</b>	10
<b>Motorcycles</b>	0
<b>Precision</b>	60%
<b>Recall</b>	88.2%
<b>d</b>	0.0617

As with clip 2, this clip shows better results than the first one, and better results the higher the resolution. However, the improvement is not quite as noticeable.

## Experimental results

Table 4.53: Running times for clip 3

Resolution	Time (s)
360p	21
480p	29
720p	89

### 4.4.4 Clip 4

The results for clip 4 are in tables 4.54, 4.58, 4.62 and 4.66.

Table 4.54: Results for clip 4 in 360p

Table 4.55: Vehicle count

	1	2	3
1	0	0	2
2	0	0	0
3	2	0	0

Table 4.56: Vehicle distribution

	1	2	3
1			50%
2			
3	50%		

Table 4.57: Measurements for clip 4 in 360p

True positives	4
False negatives	7
False positives	0
Precision	100%
Recall	36.36%
d	0.1728

Table 4.58: Results for clip 4 in 480p

Table 4.59: Vehicle count

	1	2	3
1	0	1	2
2	0	0	0
3	3	1	0

Table 4.60: Vehicle distribution

	1	2	3
1		14%	29%
2			
3	43%	14%	

Table 4.61: Measurements for clip 4 in 480p

True positives	7
False negatives	4
False positives	0
Precision	100%
Recall	63.6%
d	0.0828

Table 4.62: Results for clip 4 in 720p

Table 4.63: Vehicle count

	1	2	3
1	0	1	2
2	0	0	1
3	3	1	0

Table 4.64: Vehicle distribution

	1	2	3
1		13%	25%
2			13%
3	38%	13%	

Table 4.65: Measurements for clip 4 in 720p

True positives	8
False negatives	3
False positives	0
Precision	100%
Recall	72.73%
d	0.0678

## Experimental results

Table 4.66: Running times for clip 4

Resolution	Time (s)
360p	32
480p	42
720p	140

### 4.4.5 Clip 5

The results for clip 5 are in tables 4.67, 4.71, 4.75 and 4.79.

Table 4.67: Results for clip 5 in 360p

Table 4.68: Vehicle count

	1	2	3
1	0	0	2
2	0	0	0
3	3	1	0

Table 4.69: Vehicle distribution

	1	2	3
1			33%
2			
3	50%	17%	

Table 4.70: Measurements for clip 5 in 360p

<b>True positives</b>	6
<b>False negatives</b>	11
<b>False positives</b>	0
<b>Precision</b>	100%
<b>Recall</b>	35.29%
<b>d</b>	0.1057

Table 4.71: Results for clip 5 in 480p

Table 4.72: Vehicle count

	1	2	3
1	0	0	1
2	0	0	1
3	8	1	0

Table 4.73: Vehicle distribution

	1	0	.
1			9%
2			9%
3	73%	9%	

Table 4.74: Measurements for clip 5 in 480p

<b>True positives</b>	10
<b>False negatives</b>	7
<b>False positives</b>	1
<b>Precision</b>	90.91%
<b>Recall</b>	58.82%
<b>d</b>	0.0843

Table 4.75: Results for clip 5 in 720p

Table 4.76: Vehicle count

	1	2	3
1	0	0	2
2	1	0	1
3	8	0	0

Table 4.77: Vehicle distribution

	1	2	3 ]
1			17%
2	8%		8%
3	67%		

Table 4.78: Measurements for clip 4 in 720p

<b>True positives</b>	10
<b>False negatives</b>	7
<b>False positives</b>	2
<b>Precision</b>	83.33%
<b>Recall</b>	58.82%
<b>d</b>	0.0294

Table 4.79: Running times for clip 5

Resolution	Time (s)
360p	31
480p	42
720p	135

## 4.5 Analysis of results

First and foremost, it is noticeable that motorcycles tracking success rate is very poor in source 1, as there are several dozens motorcycles traversing the intersection. The author suspects this is due to two main reasons:

1. The fact that motorcycles are physically smaller, and therefore harder to detect. This is supported by the fact that during clip 2 some motorcycles are tracked, while during clip 3 none are, and clip 3 is taken with a lower level of zoom, therefore making the vehicles physically smaller on screen (see figure 4.4). The fact that the motorcycle detection rate is also lower on lower resolutions also supports this theory, as well as the fact that motorcycles are tracked correctly in the footage from source 2, in which they occupy a much more significant area on the screen;
2. The chaotic movement of motorcycles through the intersection, sometimes even completely stopping in packs in the middle of the road, makes it difficult for the algorithm to track them from frame to frame (see figure 4.3).



Figure 4.3: A group of motorcycles stopped in the middle of the intersection (highlighted in red)

On the other hand, the car/truck/bus recall, especially for clips 2 and 3 in high resolutions, is interesting, being above 85% and coupled with precisions sometimes above 80%. Also interesting

## Experimental results

is that in clip 2 the lower resolutions have similar recalls, and these could still be useful being close to 50%, but in clip 3, the one in which vehicles are smaller on frame, the algorithm performs very poorly (below 20% true positives) for the lower resolutions. The difference between the lower resolutions and the 720p version is much more extreme. This suggests the algorithm is susceptible to failure when vehicles are below a certain size on screen.

As for clip 1, recall is poorer than for each of the other clips individually, at least in high resolution. This is due to the fact that there is a significant shift in zoom level halfway through the clip, which makes the coordinates provided by the user for the intersection limits wrong, and ruins tracking for the period during which the camera is zooming out. When this zooming period is cut out and the limits are adjusted (that is, when clips 2 and 3 are taken individually), then the results are much better. This also justifies the choice of not using the middle section of the full clip as its own individual clip.

Another interesting point to look at is the distribution of vehicles in the matrix in the ground-truth and compare it to the algorithm's results. In most cases,  $d$  is at or below 0.1. In the case of clips 1 and clips 2, it is lower for lower resolutions, whereas for clip 3 it is extremely high. On the other hand, for clips 4 and 5, the accuracy of the distributions improves directly with resolution.

It is possible that, in some cases, when the resolution is lowered, most of the noise is being eliminated, lowering the amount of false positives, and hence improving the quality of the distribution, even if the actual absolute counts are not very accurate.

It should also be noted that in most cases, even when the number of true positives is low (below 80%), their distribution seems to still be similar to that of the ground-truth, suggesting the algorithm is usable even in those conditions to get an idea of how an intersection works. Additionally it should also be noted that the average precision is 87.5%, supporting the claim that the algorithm is robust against false positives.

Overall, one can say the algorithm tracks cars/trucks/buses successfully if there are no changes in perspective and the resolution is high enough, and that it provides useful data regarding how many vehicles move between each entrance in the intersection, even when it doesn't track all vehicles. It is also possible to say that if the size of vehicles on screen is sufficiently large, they will most likely be tracked successfully.

Because the algorithm was developed based on a set of clips, but tested successfully with some other additional ones, it is safe to assume that it is versatile enough to handle different configurations of road and different traffic patterns, and is not limited to just the example used for development.



## Experimental results



Figure 4.4: Difference in level of zoom between clips 2 (top), 3 (middle) and 4/5 (bottom)

## Experimental results



## Chapter 5

# Conclusion

An algorithm has been developed by looking into existing computer vision techniques and existing work in the field of traffic monitoring, experimenting with several algorithms and techniques and settling on those which seemed to provide results. Even though the input data used has some camera shake, which is impossible to completely eliminate from a drone-based solution, the algorithm is capable of compensating for that.

The algorithm has been tested against several clips with different characteristics and at different resolutions. It has proven capable of tracking vehicles such as cars, buses and trucks. The algorithm's accuracy as far as the absolute vehicle count benefits from higher resolution input data, but running time naturally increases the higher the resolution is. On the other hand, the accuracy of the distribution measurement sometimes remains acceptable or even improves with lower resolutions.

The tracking of motorcycles is considerably more complicated and mostly not successful if they do not occupy a sufficiently large proportion of the screen.

Based on the data extracted by the algorithm, models of the intersection being analysed can be built, both showing the absolute number of vehicles and the relative distribution of traffic flow between pairs of entrances/exits. Both of these are accurate under at least some conditions.

There are, as should be expected, areas which can be improved upon. Future work could focus on the following aspects:

- Improving the detection rate for motorcycles - this would most likely involve developing an heuristic which keeps track of when a motorcycle has been detected and attempts to predict where it will be even if it disappears between two non-consecutive frames;
- Improving the performance of the algorithm - performance could be improved by either altering the background subtraction model, or by not relying on background subtraction at all. This second option ties into the next point;
- Replace background subtraction - due to the instability of a camera on a drone, the usage of background subtraction raises inherent problems. If a suitable camera sway compensation technique could be found (for example, if there were a way to extrapolate the camera's

## Conclusion

movement in 3D), then one could forego background subtraction, and use some heuristic to identify vehicles (by detecting rectangle-like structures within a certain area);

- Removing the need for the user to specify the limits by hand - this is a different computer vision problem but it ties into the current work. It would be interesting to have the algorithm automatically detect the limits of the intersection. This could be done, in some cases, by detecting the zebra crossing;
- Use lower resolution distribution results as a guide - given that the test data suggests the algorithm gets better vehicle distribution results with lower resolution footage, it could be interesting to first do a pass against a low-res version of the input data, and use those results to then guide the search for vehicles in the higher resolution version;
- Analyse the algorithms behaviour when its parameters are changed.

# References

- [AFo] AForge.NET. Aforge.net. Available at <http://www.aforgenet.com/>, last accessed on 06/02/2015.
- [Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 8 no. 6, pages 679–714, November 1986.
- [CBMM98] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C* 6, pages 271–288, 1998.
- [CGPP03] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pages 1337–1342, October 2003.
- [CMM<sup>+</sup>06] Benjamin Coifman, Mark McCord, Rabi G. Mishalani, Michael Iswalt, and Yuxiong Ji. Roadway traffic monitoring from an unmanned aerial vehicle. *Intelligent Transport Systems, IEE Proceedings, Vol. 153, Issue 1*, pages 11–20, March 2006.
- [Del] Patrice Delmas. Morphological image processing. Available at <https://www.cs.auckland.ac.nz/courses/compsci773slc/lectures/ImageProcessing-html/topic4.htm>, last accessed on 05/02/2015.
- [Der87] Rachid Deriche. Using canny’s criteria to derive a recursively implemented optimal edge detector. *Int. J. Computer Vision, Vol. 1*, pages 167–187, April 1987.
- [DZM<sup>+</sup>07] Hongli Deng, Wei Zhang, Eric Mortensen, Thomas Dietterich, and Linda Shapiro. Principal curvature-based region detector for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition.*, 2007.
- [Eff00] Nick Efford. *Digital Image Processing: A Practical Introduction Using Java<sup>TM</sup>*. Pearson Education, 2000.
- [Far03] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. *Lecture Notes in Computer Science Volume 2749*, pages 363–370, 2003.
- [figa] Available at <http://www.ee.surrey.ac.uk/CVSSP/demos/corners/blox-css.jpg>, last accessed on 17/02/2015.
- [figb] Available at [https://hal.inria.fr/inria-00440407/file/ben\\_525\\_526\\_HOG.png](https://hal.inria.fr/inria-00440407/file/ben_525_526_HOG.png), last accessed on 17/02/2015.

## REFERENCES

- [figc] Available at [http://vip.bu.edu/files/2010/02/BU\\_Bridge\\_cars.gif](http://vip.bu.edu/files/2010/02/BU_Bridge_cars.gif), last accessed on 17/02/2015.
- [GMG12] Andrew B. Godbehere, Akihiro Matsukawa, and Ken Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. *American Control Conference (ACC)*, 2012, pages 4305–4312, June 2012.
- [Guy08] Stephen J. Guy. Available at <https://wwwx.cs.unc.edu/~sjguy/CompVis/Features/>, last accessed on 17/02/2015, 2008.
- [Hoi] Derek Hoiem. Feature tracking and optical flow. Available at [https://courses.engr.illinois.edu/cs543/sp2012/lectures/Lecture%2008%20-%20Feature%20Tracking%20and%20Optical%20Flow%20-%20Vision\\_Spring2012.pdf](https://courses.engr.illinois.edu/cs543/sp2012/lectures/Lecture%2008%20-%20Feature%20Tracking%20and%20Optical%20Flow%20-%20Vision_Spring2012.pdf), last accessed on 04/02/2015.
- [HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151. University of Manchester, 1988.
- [HYCH06] Jun-Wei Hsieh, Shih-Hao Yu, Yung-Sheng Chen, and Wen-Fong Hu. Automatic traffic surveillance system for vehicle tracking and classification. *IEEE Transactions on intelligent transportation systems vol. 7 no. 2*, June 2006.
- [Ini85] Rafael Inigo. Traffic monitoring and control using machine vision: a survey. *IEEE Trans. Ind. Electron.*, August 1985.
- [Ini89] Rafael Inigo. Application of machine vision to traffic monitoring and control. *Vehicular Technology, IEEE Transactions on Vol. 38, Issue 3* ), pages 112–122, August 1989.
- [IVT] IVT. Ivt - integrating vision toolkit. Available at <http://ivt.sourceforge.net/>, last accessed on 06/02/2015.
- [Kü08] Reinhart D. Kühne. Greenshields’ legacy highway traffic. In *75 Years of the Fundamental Diagram for Traffic Flow Theory*, pages 3–10, 2008.
- [Kau] Aditya Kaushik. Frame grabbed from the video at <http://www.mathworks.com/videos/introduction-to-kalman-filters-for-object-tracking-79674.html>, last accessed on 25/06/2015.
- [KB01] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. In *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, pages 135–144, September 2001.
- [Kim05] ZuWhan Kim. Realtime road detection by learning from one example. In *7th IEEE Workshops on Application of Computer Vision Vol. 1*, pages 455–460, 2005.
- [KMRV13] Konstantinos Kanistrasy, Goncalo Martinsy, Matthew J. Rutherfordz, and Kimon P. Valavanisy. A survey of unmanned aerial vehicles (uavs) for traffic monitoring. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013.

## REFERENCES

- [KWH<sup>+</sup>94] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on Vol. 4*, pages 3776–3781, 1994.
- [KWM94] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *Third European Conference on Computer Vision Stockholm, Proceedings, Vol. 1*, pages 189–196, May 1994.
- [Lin98] Tony Lindberg. Principles for automatic scale selection. Technical report, Computational Vision and Active Perception Laboratory (CVAP) Department of Numerical Analysis and Computing Science KTH (Royal Institute of Technology), August 1998.
- [Lin12] Tony Lindberg. Scale selection properties of generalized scale-space interest point detectors. Available at [http://download.springer.com/static/pdf/392/art%253A10.1007%252Fs10851-012-0378-3.pdf?auth66=1423241799\\_731dc89344e3a959b09aad91c3061eed&ext=.pdf](http://download.springer.com/static/pdf/392/art%253A10.1007%252Fs10851-012-0378-3.pdf?auth66=1423241799_731dc89344e3a959b09aad91c3061eed&ext=.pdf), last accessed on 04/02/2015, September 2012.
- [LK81] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.
- [LRB09] P. Loureiro, R.J.F. Rossetti, and R.A.M. Braga. Video processing techniques for traffic information acquisition using uncontrolled video streams. In *12th International IEEE Conference on Intelligent Transportation Systems (ITSC '09)*, pages 127–133. IEEE, October 2009.
- [LV01] B.P.L. Lo and S.A. Velastin. Automatic congestion detection system for underground platforms. In *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, pages 158–161, May 2001.
- [Mat] Mathworks. Computer vision system toolbox. Available at <http://de.mathworks.com/products/computer-vision/>, last accessed on 06/02/2015.
- [McL10] Jon McLoone. Edge detection applied to a photograph. Available at <http://en.wikipedia.org/wiki/File:EdgeDetectionMathematica.png>, last accessed on 17/02/2015, 2010.
- [MCUP02] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. of British Machine Vision Conference*, pages 384–396, 2002.
- [Mor80] Hans Moravec. Obstacle avoidance and navigation in real world by a seeing robot rover. Technical report, Carnegie-Mellon University Robotics Institute, September 1980.
- [odta] opencv dev team. Background subtraction. Available at [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_video/py\\_bg\\_subtraction/py\\_bg\\_subtraction.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html), last accessed on 05/02/2015.
- [odtb] opencv dev team. Optical flow. Available at [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_video/py\\_lucas\\_kanade/py\\_lucas\\_kanade.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html), last accessed on 04/02/2015.

## REFERENCES

- [Ope] OpenCV. Opencv. Available at <http://opencv.org/>, last accessed on 06/02/2015.
- [Pic04] Massimo Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics 4*, pages 3099–3104, October 2004.
- [PRK11] L.S. Passos, R.J.F. Rossetti, and Z. Kokkinogenis. Towards the next-generation traffic simulation tools: a first appraisal. In *6th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6, June 2011.
- [Pro] The VXL Project. Vxl - c++ libraries for computer vision research and implementation. Available at <http://vxl.sourceforge.net/>, last accessed on 06/02/2015.
- [PVK07] A. Puri, K. P. Valavanis, and M. Kontitsis. Statistical profile generation for traffic monitoring using real-time uav based video data. In *Control & Automation, 2007. MED '07. Mediterranean Conference on*, pages 1–6, June 2007.
- [RE15] Rosaldo J. F. Rossetti and Ronghui Liu (Eds.). *Advances in Artificial Transportation Systems and Simulation*, 2015.
- [RFW03] Ashley Walker Robert Fisher, Simon Perkins and Erik Wolfart. Sobel edge detector. Available at <http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm>, last accessed on 03/02/2015, 2003.
- [RJ05] Roya Rad and Mansour Jamzad. Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters* 26, pages 1597–1607, 2005.
- [RLT11] R.J.F. Rossetti, R. Liu, and S. Tang. Guest editorial special issue on artificial transportation systems and simulation. *IEEE Transactions on Intelligent Transportation Systems* 2 (12), 2011.
- [ROB07] R.J.F. Rossetti, E.C. Oliveira, and A.L.C. Bazzan. Towards a specification of a framework for sustainable transportation analysis. In *13th Portuguese Conference on Artificial Intelligence*, December 2007.
- [Set97] Christopher John Setchell. *Applications of Computer Vision to Road-traffic Monitoring*. PhD thesis, University of Bristol, 1997.
- [SG99] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on Vol 2*, pages 246–252, June 1999.
- [Sha] Ian G. R. Shaw. The rise of the predator empire: Tracing the history of u.s. drones. Available at <https://understandingempire.wordpress.com/2-0-a-brief-history-of-u-s-drones/>, last accessed on 05/02/2015.
- [SSFD11] Amir Salarpour, Arezoo Salarpour, Mahmoud Fathi, and MirHossein Dezfoulan. Vehicle tracking using kalman filter and features. *Signal & Image Processing : An International Journal*, Vol. 2, No. 2, June 2011.
- [ST94] Jianbo Shi and Carlo Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.

## REFERENCES

- [TSK<sup>+</sup>82] Sadao Takaba, Masao Sakauchi, Takashi Kanelco, Byong Won Hwang, and Tomiyoshi Sekine. Measurement of traffic flow using real time processing of moving pictures. *Vehicular Technology Conference, 32nd IEEE Vol. 32*, pages 488–494, May 1982.
- [Voo87] Harry Voorhees. Finding texture boundaries in images. Technical report, Massachusetts Institute of Technology Artificial Intelligence Laboratory, June 1987.
- [WADP97] C. Wren, A. Azarhayejani, T. Darrell, and A.P. Pentland. Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions Vol. 19 Issue 7*, pages 780–785, July 1997.
- [WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Available at [http://www.cs.unc.edu/~welch/media/pdf/kalman\\_intro.pdf](http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf), last accessed on 17/02/2015, 1995.
- [Ziv04] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, pages 23–31, August 2004.
- [ZT88] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, pages 537–559, 1988.